**TurtleFrame クラス**

コンストラクタ

- **TurtleFrame()** TurtleFrame を，デフォルトの大きさ (400 × 400) で生成する。
- **TurtleFrame(int width, int height)** TurtleFrame を width × height の大きさで生成する。

メソッド

- **void add(Turtle t)** タートル t をこのウィンドウに追加する。
- **void remove(Turtle t)** タートル t をこのウィンドウから削除する。
- **void clear()** いままでに描かれたすべての線を消す。
- **void addMesh()** 方眼紙のような桝目を表示する。

**Turtle クラス**

コンストラクタ

- **Turtle()** (200,200) という座標に 0 度の角度で Turtle を生成する。
- **Turtle(int x, int y, int angle)** (x, y) という座標に angle 度の角度で Turtle を生成する。

メソッド

- **void fd(int n)** n だけ前に進む。
- **void bk(int n)** n だけ後ろに進む。
- **void rt(int n)** n 度だけ右に向きを変える。
- **void lt(int n)** n 度だけ左に向きを変える。
- **void setColor(java.awt.Color nc)** ペンの色を nc に変更する。
- **int moveTo(int x,int y)** (x, y) という座標の方向を向き，(x, y) まで進む。動いた距離を返す。
- **int moveTo(Turtle t)** タートル t の方向を向き，t と同じ座標まで進む。動いた距離を返す。
- **int moveTo(int x,int y, int angle)** (x, y) という座標の方向を向き，(x, y) まで進んだ後，angle の方向を向く。動いた距離を返す。
- **int getX()** 現在の座標の X 成分を返す。
- **int getY()** 現在の座標の Y 成分を返す。
- **int getAngle()** 現在の角度を返す。
- **Turtle clone()** 自分と同じ状態のタートルを生成して返す。
- **void up()** ペンを上げる。
- **void down()** ペンを下ろす。ペンを下ろした状態で進むと，その軌跡が画面に線として描画される。
- **boolean isDown()** ペンを上げた状態なら false，下げた状態なら true を返す。
- **void speed(int x)** タートルの動きの速さを x に設定する。x = 20 がデフォルトである。数字が小さいほど速い。
- **static void speedAll(int x)** タートル全体の速さを 1〜3 で指定する。1 が高速，3 が低速。

フィールド

- **java.awt.Color tcolor** 亀の絵の色。初期値は緑色。
- **double tscale** 亀の絵の大きさを表す浮動小数点数。初期値は 0.4。
- **static boolean withTurtleAll** もし，false ならすべてのタートルが亀の絵を表示しないで瞬時に描画を行う。true なら通常の描画を行う。初期値は true。

P.28

**Math クラス（一部）**

フィールド

- **static final double PI** 円周率（$\pi$）の値 (= 3.14159...)。
- **static final double E** 自然対数の値 (= 2.71828...)。

メソッド

- **static double sin(double r)** sin(r) を返す。r はラジアンで与える。
- **static double cos(double r)** cos(r) を返す。r はラジアンで与える。
- **static double sqrt(double r)** $\sqrt{r}$ を返す。
- **static int max(int x, int y)** x と y の最大値を返す。
- **static double max(double x, double y)** x と y の最大値を返す。
- **static double random()** 0.0 以上，1.0 未満の乱数を返す。
- **static double toRadians(double angdeg)** 度をラジアンに変換する。

P.29

**System クラス（一部）**

フィールド

- **static final java.io.PrintStream out** 標準出力につながっている PrintStream オブジェクト。
- **static final java.io.InputStream in** 標準入力につながる InputStream オブジェクト。

メソッド

- **static void exit(int status)** このプログラムの実行（すなわち，この仮想マシンの実行）を終了する。status は，正常終了のときは 0，エラーによる終了のときは 1 を指定する。

P.30

**String クラス（一部）**

メソッド

- **static String valueOf(x)** x の文字表現を返す（すべての型の x に多重定義されている）。
- **boolean equals(String str)** この文字列が str と同じ内容の文字列なら true を返す。
- **int compareTo(String str)** この文字列と str と辞書に現れる順序で比較。この文字列が前なら負の値，後ろなら正の値，両者が同じ文字列なら 0 を返す。
- **char charAt(int index)** index 番目の文字を返す。
- **int indexOf(char c)** 文字列に c が最初に現れるインデックスを返す。
- **int length()** この文字列の長さを返す。

P.59

**Integer クラス（一部）**

コンストラクタ

- **Integer(int value)** value を値とする Integer オブジェクトの生成。

メソッド

- **int intValue()** このオブジェクトの意味する int 値を返す。
- **static int parseInt(String s)** 文字列 s を 10 進表記とする int 値を返す。
- **static Integer valueOf(String s)** s を 10 進表記とする Integer オブジェクトを生成して返す。

フィールド

- **static final int MAX_VALUE** int として表現できる最大数 ($2^{31}-1$)。
- **static final int MIN_VALUE** int として表現できる最小数 ($-2^{31}$)。

P.60

**HTurtle クラス**

Turtle クラスを拡張している。

メソッド

**void polygon(int n, int s)** 現在のタートルの位置から，1 辺の長さ s の n 角形を右回りに描く。

**void house(int s)** 現在のタートルの位置から，1 辺の長さ s の家の絵を描く。

P.66

**Stepper クラス**

Turtle クラスを拡張している。

メソッド

**void step()** 1 辺の長さが size の n 角形の 1 辺だけを描く。すでに描き終っていたら何もしない。

フィールド

**int n** 多角形の辺の数。

**int size** 1 辺の長さ。

P.70

**CStepper クラス**

Turtle クラスを拡張している。

コンストラクタ

**CStepper(int n, int size, int x, int y, int angle)** n 角形を 1 辺の長さ size で描く CStepper を，(x, y) の座標に angle の角度で作成。

**CStepper(int n, int size)** n 角形を 1 辺の長さ size で描く CStepper を (200,200) の座標に 0 度の角度で作成。

メソッド

**void step()** n 角形の 1 辺だけを描く。すでに n 角形を描き終っていたら何もしない。



P.118

**ImageIcon クラス**

コンストラクタ

**ImageIcon(String filename)**
画像ファイル名を表す文字列を指定して，アイコンを作る。ファイル名は，プログラムのクラスファイルがあるディレクトリからの相対的位置を指定する。

**ImageIcon (Image image)**
Image クラスのオブジェクトからアイコンを作る。Image オブジェクトの生成については 12.4 節を参照。

P.121

**FlowLayout クラス**

コンストラクタ

**FlowLayout()**

**FlowLayout(int align)**

**FlowLayout(int align, int hgap, int vgap)**
第 1 引数では，コンテナ中のコンポーネントの揃え方(アライメント)を，FlowLayout クラスのクラス変数を使って指定する。デフォルトでは CENTER が指定される。引数 hgap と vgap はそれぞれコンポーネント間の水平，垂直方向の間隔を指定するもので，デフォルトは 5 ピクセルである。

引数 align に指定する値

**FlowLayout.LEFT** コンポーネントの行を左詰めにする。

**FlowLayout.CENTER** コンポーネントの行をコンテナの中央に置く。

**FlowLayout.RIGHT** コンポーネントの行を右詰めにする。

P.122

**BorderLayout クラス**

コンストラクタ

**BorderLayout()**

**BorderLayout(int hgap, int vgap)**
引数 hgap, vgap では，コンポーネントの水平，垂直方向の間隔を指定する。引数を指定しないと，間隔は 0 になる。

フィールド

add メソッドの第 2 引数に指定する値(領域名称)

**BorderLayout.CENTER** 中央の領域

**BorderLayout.NORTH (BorderLayout.PAGE_START)** 上端の領域。

**BorderLayout.SOUTH (BorderLayout.PAGE_END)** 下端の領域。

**BorderLayout.WEST (BorderLayout.LINE_START)** 左端の領域。

**BorderLayout.EAST (BorderLayout.LINE_END)** 右端の領域。

上記の領域名称とその意味は，文字を左から右へ横書きする言語環境が設定されている場合での(一般の日本語環境はこれにあたる)ものである。括弧の中の値も同じ意味で使えるが，両者を混合して使わないこと。

**GridLayout クラス**

コンストラクタ

**GridLayout()**
**GridLayout(int rows, int cols)**
**GridLayout(int rows, int cols, int hgap, int vgap)**
コンポーネントを配置する行数または列数を引数 rows と cols で，コンポーネント間の水平，垂直方向の間隔を hgap と vgap で指定する。
間隔のデフォルト値は 0 である。rows と cols のどちらかは必ず非ゼロでなくてはならない。
引数がないコンストラクタは，1 行で，間隔が 0 の GridLayout のオブジェクトを作る。



**BoxLayout クラス**

コンストラクタ

**BoxLayout(Container container, int axis)**
第 1 引数にコンポーネントを配置するコンテナを，第 2 引数にどの方向に並べるかを指定する。
BoxLayout を設定するコンテナが自分自身の場合は，第 1 引数に this を指定する。第 2 引数には，BoxLayout.Y_AXIS（縦方向に並べる）か BoxLayout.X_AXIS（横方向に並べる）かを指定する。
例： setLayout(**new** BoxLayout(**this**, BoxLayout.Y_AXIS));



**JLabel クラス**

コンストラクタ

**JLabel()** 中身のないラベルを生成。
**JLabel(Icon icon)**
**JLabel(String text)**
**JLabel(String text, Icon icon, int align)**
icon で表示するアイコンを，text で表示する文字列を指定する。align はアイコンと文字のラベル内水平方向の配置位置で，SwingConstants インターフェースの定数(LEFT，CENTER，RIGHT)を使って指定する。デフォルトでは中央揃え。

メソッド

**void setText(String text)** ラベルに表示する文字列を設定する。
**void setIcon(Icon icon)** ラベルに表示するアイコンを設定する。



**AbstractButton クラス**

メソッド

**void setText(String text)** ボタンに表示する文字列を設定。
**void setIcon(Icon icon)** ボタンに表示するアイコンを設定。
**void setPressedIcon(Icon icon)** ボタンが押されたときの画像を設定。
**void setSelectedIcon(Icon icon)** ボタンが選択されたときの画像を設定。
**boolean isSelected()** ボタンの選択状態を返す。
**void setSelected(boolean b)** 選択状態 (true) か非選択状態 (false) かを設定。
**void addActionListener(ActionListener l)** ActionEvent リスナーを設定。
**void addChangeListener(ChangeListener l)** ChangeEvent リスナーを設定。
**void addItemListener(ItemListener l)** ItemEvent リスナーを設定。
**void setActionCommand(String command)**
ボタンに対する処理に名前(コマンド名)を設定。イベント処理に使う。



**JButton クラス**

コンストラクタ

**JButton()** 何も表示しないボタンを生成。
**JButton(Icon icon)** アイコン付きのボタンを生成。
**JButton(String text)** 文字列付きのボタンを生成。
**JButton(String text, Icon icon)** 文字列とアイコンが付いたボタンを生成。



**JCheckBox クラス**

コンストラクタ

**JCheckBox()**
**JCheckBox(Icon icon)**
**JCheckBox(String text)**
**JCheckBox(String text, Icon icon)**
**JCheckBox(Icon icon, boolean state)**
**JCheckBox(String text, boolean state)**
**JCheckBox(String text, Icon icon, boolean state)**

text で文字列を，icon で非選択時の画像を指定する。state で初期状態で選択するかどうかを指定。デフォルトは false (非選択)。引数のないコンストラクタは，デフォルトのアイコンがついた文字列のないチェックボックスを生成。

## JComboBox クラス

コンストラクタ

JComboBox() 空の JComboBox を生成。

JComboBox(Object[] items) 配列要素を項目とする JComboBox を生成。配列オブジェクトの文字列表現(toString メソッドの戻り値)が表示される。

メソッド

void addItem(Object object) リスト項目に object を追加。

void insertItemAt(Object object, int index) index で指定した位置に object を挿入する。先頭は 0。

void setMaximumRowCount(int count) メニューに表示する最大の項目数を設定。項目の和がこれを越えると，スクロールバーが付く。

int getSelectedIndex() 選択されている項目の番号を返す。先頭は 0。

Object getSelectedItem() 選択されている項目を返す。

Object getItemAt(int index) 指定された番号の項目を返す。先頭は 0。

void setSelectedIndex(int index) 指定した番号の項目を選択する。

void setSelectedItem(Object object) 指定した項目を選択する。

void setEditable(boolean b) 表示域が編集可能かどうかを指定。true は編集可。デフォルトは不可。

P.130

## JPanel クラス

コンストラクタ

JPanel() FlowLayout で配置するパネルを生成。

JPanel(LayoutManager layout) レイアウトマネージャを指定して，パネルを生成。

Container クラスから継承したメソッド

void setLayout(LayoutManager l) レイアウト方式を設定する。

void add(Component c) パネルにコンポーネントを配置する。

void add(Component c, int n) add メソッドの引数はレイアウト方式によって異なる。

void add(Component c, Object o)

Component getComponent(int n) パネル内の n 番目のコンポーネントを返す。

Component getComponentAt(int x, int y) 指定した座標位置にあるコンポーネントを返す。



## JFrame クラス

コンストラクタ

JFrame() タイトルのないウィンドウを生成。

JFrame(String title) title でウィンドウのタイトルを指定する。

メソッド

Container getContentPane() コンテントペインを得る。

void setContentPane(Container content) コンテントペインを設定する。

void setDefaultCloseOperation(int operation)
ウィンドウで「閉じる」操作をした場合の処理を設定する。引数に指定する値は下のいずれか。
JFrame.EXIT_ON_CLOSE ... System の exit メソッドを使用してプログラムを終了する。
WindowConstants.DO_NOTHING_ON_CLOSE ... 何もしない。
WindowConstants.HIDE_ON_CLOSE (デフォルト) ... フレームを隠す。
WindowConstants.DISPOSE_ON_CLOSE ... フレームを隠して破棄(プログラム終了)。

void setLayout(LayoutManager l) レイアウト方式を設定する。

void setTitle(String str) フレームのタイトルを設定する。

void pack() フレームの大きさを必要最小限の大きさに変更する。

void setSize(int w, int h) 大きさを幅 (w)，高さ (h) に変更する。

void setSize(Dimension d) 大きさを幅 (d.width)，高さ (d.height) に変更する。

void setVisible(boolean b) 表示 (true) するか，しない (false) かを設定する。

void setJMenuBar(JMenuBar bar) メニューバーを設定する。

P.132

## JComponent の色とフォントの設定メソッド

void setBackground(Color c) 背景色を設定する。

void setForeground(Color c) 描画色を設定する。

void setFont(Font f) 使用する文字のフォントを設定する。

P.134

## JComponent の境界線設定メソッド

void setBorder(Border border) 描画する境界線を設定する。

p.138

**Component クラスの大きさ/位置を返すメソッド**

**int getWidth()** 幅を返す。
**int getHeight()** 高さを返す。
**Point getLocation()** 外側(下)のコンポーネント座標空間での左上角の位置を返す。Point は (x, y) 座標空間での位置を表すクラス。

p.138

**Compoment クラスの描画メソッド**

**void repaint()** コンポーネント全体を再描画する。
**void repaint(int x, iny y, int w, int h)** 引数で指定されたコンポーネント上の矩形領域を再描画する。

p.141

**Graphics クラスの色とフォントの設定メソッド**

■色
**void setColor(Color c)**
描画色を設定。新たに setColor を実行しない限り，ここで指定した色が以後の描画に使われる。

■フォント
**void setFont(Font f)**
文字の描画に使うフォントを設定する。新たに setFont を実行しない限り，ここで指定したフォントが以後の描画に使われる。



p.139

**Graphics クラスの描画メソッド**

■直線
**void drawLine(int x1, int y1, int x2, int y2)** (x1,y1) と (x2,y2) の間に線を描く。

■四角
**void drawRect(int x, int y, int width, int height)**
左上角が (x,y) で，幅 width，高さ height の矩形の輪郭を描く。
**void fillRect(int x, int y, int width, int height)**
左上角が (x,y) で，幅 width，高さ height の矩形の領域を描画色で塗りつぶす。
**void clearRect(int x, int y, int width, int height)**
左上角が (x,y) で，幅 width，高さ height の矩形の領域を背景色で塗りつぶす。
**void draw3DRect(int x, int y, int width, int height, boolean raised)**
3 次元の矩形輪郭を描く。左上から光をあてたように輪郭が強調される。raised を true とすると浮き出た輪郭になる。
**void fill3DRect(int x, int y, int width, int height, boolean raised)**
3 次元の矩形領域を塗りつぶす。左上から光をあてたように輪郭が強調される。raised を true とすると浮き出た輪郭になる。
**void drawRoundRect(int x, int y, int width, int height, int arcW, int arcH)**
角の丸い矩形輪郭を描く。arcW，arcH は角の丸みの水平方向，垂直方向の半径である。図 12.2 は丸みをともに 10 と指定した例。
**void fillRoundRect(int x, int y, int width, int height, int arcW, int arcH)**
角の丸い矩形領域を描画色で塗りつぶす。

■楕円
**void drawOval(int x, int y, int width, int height)**
左上角が (x,y) で，幅 width，高さ height の矩形に接した楕円の輪郭を描く。
**void fillOval(int x, int y, int width, int height)**
左上角が (x,y) で，幅 width，高さ height の矩形に接した楕円領域を描画色で塗りつぶす。

■円弧
**void drawArc(int x, int y, int width, int height, int startA, int arcA)**
左上角が (x,y) で，幅 width，高さ height の矩形に接した円弧を描く。円弧の中心はその矩形の中心になる。startA は円弧の始まる位置。時計の 3 時位置を 0 度とし，反時計回りが正の値，時計回りが負の値になる (90 は 12 時の位置，−90 は 6 時の位置から円弧を始める意味)。arcA は描く円弧の角度。正の値を指定すると startA の位置から反時計回りで，負の値を指定すると時計回りで，指定した角度の円弧ができる。図 12.2 は startA に 45，arcA に 225 を指定した結果である。
**void fillArc(int x, int y, int width, int height, int startA, int arcA)**
パイ型の領域を塗りつぶす。引数の意味は drawArc と同じ。

■折れ線
**void drawPolyline(int[] xPoints, int[] yPoints, int n)**
折れ線を描く。n で頂点の数を，各点を x 座標の配列 xPoints と y 座標の配列 yPoints で指定する。

■多角形
**void drawPolygon(int[] xPoints, int[] yPoints, int n)**
多角形を描く。多角形の頂点の数は n で，頂点は x 座標の配列 xPoints と y 座標の配列 yPoints で指定する。最初の点と最後の点が違う場合，その間に線を引き図形を自動的に閉じる。
**void drawPolygon(Polygon p)** 多角形を描く。多角形は Polygon オブジェクトで指定する。
**void fillPolygon(int[] xPoints, int[] yPoints, int n)** 描画色で塗った多角形を描く。
**void fillPolygon(Polygon p)** 描画色で塗った多角形を描く。

■文字
**void drawString(String str, int x, int y)**
Graphics オブジェクトのフォントと文字を使って，str で指定された文字列を描く。x と y は最初の文字のベースラインの左端の座標である。

---

Polygon クラスは閉じた 2 次元領域(多角形)の情報をカプセル化したクラス。頂点の x 座標の配列 xPoints と y 座標の配列 yPoints，頂点数を指定して生成する。最初の点と最後の点が線で結ばれて閉じる。例：new Polygon(xPoints, yPoints, n)。

P.162

(11)

**表 13.1　イベントの種類とリスナーインターフェース**

| | イベントクラス | イベント発生のタイミング | リスナーインターフェース |
|---|---|---|---|
| AWT | ActionEvent | ボタンをクリックしたとき，メニューを選択したとき，テキストフィールド上で Enter(Return) キーを押したとき | ActionListener |
| | AdjustmentEvent | スクロールバーの値が変化したとき | AdjustmentListener |
| | ComponentEvent | コンポーネントの大きさ，位置，表示が変化したとき | ComponentListener |
| | ContainerEvent | コンテナにコンポーネントが追加，削除されたとき | ContainerListener |
| | FocusEvent | コンポーネントがキーボードフォーカスを取得あるいは失ったとき | FocusListener |
| | ItemEvent | 項目が選択，選択解除されたとき | ItemListener |
| | KeyEvent | キー入力があったとき | KeyListener |
| | MouseEvent | コンポーネント内でマウス操作があったとき | MouseListener (クリック，マウスの出入り)<br>MouseMotionListener (マウスの移動，ドラッグ)<br>MouseInputListener (上記二つを合わせたもの)* |
| | WindowEvent | ウィンドウの状態が変化したとき | WindowListener |
| Swing | CaretEvent | テキストコンポーネント内でキャレットが移動したとき，選択範囲が変化したとき | CaretListener |
| | ChangeEvent | コンポーネントの状態が変化したとき | ChangeListener |
| | ListSelectionEvent | リストの選択範囲が変化したとき | ListSelectionListener |

* MouseInputListener は Swing のインターフェースである。
AWT のイベント処理は java.awt.event パッケージに，Swing は javax.swing.event パッケージに定義されている。

P.165

**表 13.3　イベントの情報を得るメソッド**

| メソッド | 使　い　方 |
|---|---|
| **EventObject クラス**<br>Object getSource() | イベントの起こったオブジェクトを返す。複数のイベントソースに同じリスナーが登録されているとき，どのオブジェクトで起こったイベントかを区別するのに使う。 |
| **ActionEvent クラス**<br>String getActionCommand() | ActionEvent が起こったコンポーネントのコマンド名を String として返す。<br>コマンド名とは，AbstractButton のサブクラス，JComboBox，JTextField に対して setActionCommand メソッドを使って指定できる文字列のこと。同じ動作をする複数のコンポーネントのイベント処理をまとめて記述するような場合に使う。 |
| **AdjustmentEvent クラス**<br>int getValue() | Adjustable コンポーネント(スクロールバー)の値を int として返す。 |
| **ItemEvent クラス**<br>Object getItem()<br>int getStateChange() | 変化があったオブジェクトを返す。<br>項目が選択されたのか選択解除されたのかを，ItemEvent のクラス変数 SELECTED または DESELECTED で返す。 |
| **KeyEvent クラス**<br>char getKeyChar()<br>int getKeyCode() | イベントのキーの文字を返す。<br>イベントのキーの整数型の文字コードを返す。 |
| **MouseEvent クラス**<br>int getX()<br>int getY()<br>int getButton() | マウス操作があった場所の x 座標を返す。<br>マウス操作があった場所の y 座標を返す。<br>操作されたマウスボタンを MouseEvent のクラス変数 BUTTON1, BUTTON2, BUTTON3, NOBUTTON のいずれかで返す。 |
| **ListSelectionEvent クラス**<br>int getFirstIndex()<br>int getLastIndex() | 選択に変更があったリスト項目の最初の番号を返す。<br>選択に変更があったリスト項目の最後の番号を返す。 |

P.164

(12)

**表 13.2　リスナーインターフェースとイベント処理メソッド**

| リスナーインターフェース | イベント処理メソッド | 対応するアダプター* | ソースとなる Swing コンポーネント |
|---|---|---|---|
| ActionListener | actionPerformed | — | AbstractButton のサブクラス**,<br>JComboBox, JTextField, |
| AdjustmentListener | adjustmentValueChanged | — | JScrollBar |
| ComponentListener | componentMoved<br>componentResized<br>componentShown<br>componentHidden | ComponentAdapter | Component のサブクラス |
| ContainerListener | componentAdded<br>componentRemoved | ContainerAdapter | Container のサブクラス |
| FocusListener | focusGainded<br>focusLost | FocusAdapter | Component のサブクラス |
| ItemListener | itemStateChanged | — | AbstractButton のサブクラス<br>JComboBox |
| KeyListener | keyPressed<br>keyReleased<br>keyTyped | KeyAdapter | Component のサブクラス |
| MouseListener | mouseClicked<br>mouseEntered<br>mouseExited<br>mousePressed<br>mouseReleased | MouseAdapter | Component のサブクラス |
| MouseMotionListener | mouseDragged<br>mouseMoved | MouseMotionAdapter | Component のサブクラス |
| WindowListener | windowActivated<br>windowClosed<br>windowClosing<br>windowDeactivated<br>windowDeiconfied<br>windowIconfied<br>windowOpened | WindowAdapter | Window のサブクラス |
| CaretListener | caretUpdate | — | JTextComponent のサブクラス |
| ChangeListener | stateChanged | — | AbstractButton のサブクラス<br>JSlider, JSpinner,<br>JProgressBar, JTabbedPane |
| ListSelectionListner | valueChanged | — | JList |

*アダプターについては 13.4.2 項で説明する。
**AbstractButton のサブクラスには JButton, JCheckBox, JMenu, JMenuItem, JRadioButton などがある。

P.179

**JCompoment クラスのキーボードアクション関連メソッド**

**void registerKeyboardAction(ActionListener anAction, String aCommand, KeyStroke aKeyStroke, int aCondition)**
**void registerKeyboardAction(ActionListener anAction, KeyStroke aKeyStroke, int aCondition)**

**anAction:** アクションリスナー。
**aCommand:** アクションイベントのコマンド名。
**aKeyStroke:** キーボードからの入力。
**aCondition:** アクションイベントを起こすキーフォーカスの場所。下のいずれかの値を指定。
**JComponent.WHEN_FOCUSED** 自分にフォーカスがあるとき
**JComponent.WHEN_IN_FOCUSED_WINDOW** 自分あるいは同じウィンドウにフォーカスがあるとき
**JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT** 自分あるいは親のコンポーネントにフォーカスがあるとき

P.208

(13)

**Scanner クラス**

コンストラクタ

**Scanner(File source) throws FileNotFoundException**
**Scanner(InputStream source)**

**Scanner(Readable source)**
引数に，ファイル，入力ストリーム，Reader クラスのサブクラスのオブジェクトを指定し，そこからデータを読む Scanner オブジェクトを生成する。
**Scanner(String source)** 指定された文字列からデータを読む Scanner を生成する。

メソッド

**void close()** Scanner を閉じる。Scanner がすでに閉じている場合は，何もしない。
**String findInLine(String pattern)** 区切り文字を無視して，引数のパターンと一致するデータを検索し，文字列として返す。
**boolean hasNext()** 入力に次のトークンがある場合は true を返す。
**boolean hasNextDouble()** 入力の次のトークンが double 型の値として解釈可能な場合に true を返す。
**boolean hasNextInt()** 入力の次のトークンが int 型の値として解釈可能な場合に true を返す。
**String next()** 次のトークンを読んで文字列として返す。
**double nextDouble()** 入力の次のトークンを double 型として読む。
**int nextInt()** 入力の次のトークンを int 型として読む。
**Scanner skip(String pattern)** 引数のパターンに一致する入力をスキップする。
**Scanner useDelimiter(String p)** 区切り文字のパターンを文字列として指定する。
**Scanner useLocale(Locale locale)** ロケールを設定する。Locale オブジェクトは，特定の地理的，文化的地域を表すためのもので，日本の場合は Locale.JAPAN を指定。Scanner がデータを解析する際に影響する。

P.211

**表 15.3** 書式付きの出力を行うメソッド

| クラス | メソッド |
|---|---|
| PrintStream<br>PrintWriter | printf(String format, Object... args)<br>format(String format, Object... args)<br>第 1 引数に書式を表す文字列，第 2 引数以降に出力する値を指定する。<br>第 1 引数の指示に従い書式化された文字列を出力ストリームに書き込む。<br>戻り値はそのストリーム自身(PrintStream あるいは PrintWriter)。 |
| String | static String format(String format, Object... args)<br>第 1 引数に書式を表す文字列，第 2 引数以降に出力する値を指定する。<br>第 1 引数の指示に従い書式化された文字列を返す。 |